DATE: 31 January 19

SOFTWARE SPECIFICATION

for

KA630-A CONSOLE PROGRAM

REVISION 2.2

Written By:	A. Randall Heap	MLO5-5/E71
Issued By:	A. Randall Heap	
Reviewed By:	Jesse Lipcon	MLO5-5/E71
	Jay Nichols	MLO5-5/E71
	Kathy Morse	ZK01-1/D42

C O M P A N Y C O N F I D E N T I A L

COPYRIGHT (c) 1983 BY
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

DATE: 31 January 19

SOFTWARE SPECIFICATION

for

KA630-A CONSOLE PROGRAM

REVISION 2.2

Written By:	A. Randall Heap	MLO5-5/E71
Issued By:	A. Randall Heap	
Reviewed By:	Jesse Lipcon	MLO5-5/E71
	Jay Nichols	MLO5-5/E71
	Kathy Morse	ZK01-1/D42

C O M P A N Y C O N F I D E N T I A L

COPYRIGHT (c) 1983 BY
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

PREFACE

This is a Software Product Specification. It is the definitive description in measurable terms of the goals, capabilities, and external characteristics of the KA630 console program. It is the commitment of WHAT IS TO BE BUILT as agreed by the Product Team.

No changes are to be made to the product as described here, unless approved and documented as an amendment to this Specification.

Associated Documents:

1. The KA630 Processor Specification which describes the processor board of which this software product (in ROM) is a component.

2. The KA630 System Development Plan, which describes the development project to design, build, test, evaluate, and deliver this product.

3. The KA630 ROM Diagnostic Specification which describes the ROM based diagnostics which are part of this program.

4. DEC Standard 32, the VAX Architecture Standard, which includes the definition of the VAX console system (13-Sep-82 - Rev 3).

5. The VIDEO SYSTEM STANDARDS Reference Manual which defines the video terminal architecture for DEC terminals.

6. DEC Standard 138, "Registry of Control Functions for Character Imaging Devices".

Change History:

Date: 25-July-1983

Revision 1.0 Original KA630 phase zero version.

Date: 22-February-1984

Revision 1.15 Revisions for phase 1
VMB details and figures defined according to "MicroVAX I CPU Technical Description", second draft, October 1983. This revision brings the specification up to phase 1 status. All subsequent revisions will be tracked by change bars in the left margin.

Revision 1.16 Revisions for phase 1 - continued
Spelling corrections, deleted vestigial text fragments

remaining from previous versions, a correction to th discussion of national replacement characters was made powerup mode was simplified, QVSS location wa described, boot flag bits 0 and 7 were dropped.

Revision 1.17 Revisions for phase 1 - continued

Power-up modes redefined, section on diagnostic enlarged, number of console languages enlarged from 5 t 10, documentation of all TOY registers strictly interna to the console program was removed, console flag bit moved to CPMBX TOY register, CPFLG TOY registe eliminated, power-up figures revised.

Revision 1.18 Revisions for phase 1 - continued

Revision of appendix a, LED and console output codes halt message texts converted to abbreviation so as t avoid requirements for translation.

Date: 22-Jun-1984

Revision 1.19 Revisions for delayed phase 1 - continued

Discussion of the power-up sequence and the overal organization of the beginning of the Softwar Capabilities section was revised. Wording around dis boot was modified.

Cleanup for a pre-phase-1 release.

Date: 18-Jul-1984

Revision 1.20 Converted processor name from KDQ32 to KA630, adde Portuguese to the list of supported languages.

Date: 2-Aug-1984

Revision 1.21 Changed spec name from KA630 to KA630-A. Delete vestigial documentation of "interrupt page" base console memory.

Date: 10-Sept-84

Revision 2.0 Text of user-friendly console messages revised an reduced. Post Phase 1 copy.

Date: 2-Nov-84

Revision 2.1 Rearrange test sequence, add QxSS information, chan
screen displays.

Date: 31 January 1985

Revision 2.2 Correct PROM boot definition. Correct language defau
alg. displays.

CONTENTS

1 PRODUCT SUMMARY 1
2 ENVIRONMENT 1
2.1 Users 1
2.2 Hardware 2
2.3 Software 2
2.4 Services 2
3 SOFTWARE CAPABILITIES 2
3.1 Power-up 3
3.1.1 Power-up Mode 3
3.1.2 Power Stabilization And ROM Checksum Checks . . 4
3.1.3 Console Program Initialization 4
3.1.4 Battery Backup Check 6
3.1.5 Test Interprocessor Communication Register . . . 6
3.1.6 Determining The Console Terminal Type 6
3.1.6.1 QxSS Determination 6
3.1.6.2 Console Terminal Determination 7
3.1.7 Console Message Language Check 7
3.2 Entry/Dispatch 9
3.3 Diagnostics 10
3.4 Restart 11
3.5 Bootstrap 12
3.5.1 Primary Bootstrap Program (VMB) 15
3.5.1.1 Bootstrap Devices 16
3.5.1.2 Bootstrap Command Fl 16
3.5.1.3 Booting From Disk 17
3.5.1.4 Booting From Tape 19
3.5.1.5 Booting From PROM 20
3.5.1.6 Booting From DEQNA 20
3.5.1.7 Booting An Auxiliary Processor 21
3.5.2 Secondary Bootstrap Program 22
3.6 Console I/O Mode (System Halted) 25
3.6.1 Console Control Characters 25
3.6.2 Console Command Syntax 27
3.6.3 Console Command Keywords 27
3.6.4 References To Processor Registers And Memory . 28
3.6.5 Console Commands 28
3.6.5.1 BOOT 28
3.6.5.2 CONTINUE 29
3.6.5.3 DEPOSIT 29
3.6.5.4 EXAMINE 31
3.6.5.5 FIND 32
3.6.5.6 INITIALIZE 32
3.6.5.7 HALT 32
3.6.5.8 REPEAT 33
3.6.5.9 START 33
3.6.5.10 TEST 33
3.6.5.11 UNJAM 33
3.6.5.12 Binary Load And Unload Command 34
3.6.5.13 Comment 35
3.6.6 Console Errors And Error Messages 35
3.6.7 Halts And Halt Messages 36

3.7	Program I/O Mode (System Running)	37
4	PUBLICATIONS	38
5	PACKAGING	38
6	INSTALLABILITY	38
7	EASE OF USE	38
8	PERFORMANCE	40
9	RELIABILITY	40
10	MAINTAINABILITY	41
11	MAINTENANCE	41
12	COMPATIBILITY	41
12.1	Product Compatibility	42
12.2	Standards Conformance	42
12.3	Internationalization	42
13	EVOLVABILITY	43
14	COSTS	43
15	TIMELINESS	43
16	CONSTRAINTS AND TRADEOFFS	44

APPENDIX A INTERPRETATION OF KA630 LED AND CONSOLE DISPLAY CODES

APPENDIX B SUMMARY OF I/O REGISTERS USED BY CONSOLE PROGRAM

B.1	BOOT AND DIAGNOSTIC CONFIGURATION REGISTER (BDR)	B-1
B.2	TIME OF YEAR (TOY) CLOCK REGISTERS	B-2
B.2.1	Console Program Mailbox (CPMBX)	B-2
B.3	SYSTEM IDENTIFICATION EXTENSION REGISTER (SIDEX)	B-3
B.4	CONSOLE TERMINAL REGISTERS	B-4
B.5	INTERPROCESSOR COMMUNICATION REGISTER (IPCR)	B-4

APPENDIX C QXSS AND APT

APPENDIX D OUTSTANDING ISSUES

FIGURES

1	Console Memory Map	5
2	User Language Prompt	8
3	QxSS Keyboard prompts	9
4	RPB Format	12
5	Bootblock Format	18
6	Extended RPB	23
7	Secondary Bootstrap Argument list	24
8	Secondary Bootstrap Memory Map	24
9	Console Display on Successful Power-Up	39
10	Console Display on Power-Up with Hard Errors	40

TABLES

1	Power-up Modes	3
2	Console Entry Decision Table	10
3	VMB Register Usage	14
4	VMB Bootstrap Command Flags	16
5	Console Error Messages	35
6	KA630 Halt Messages	37
A-1	KA630 LED Interpretation	A-1

1 PRODUCT SUMMARY

This document describes the KA630 console program. This program conjunction with KA630 hardware implements the KA630 console syst which gains control whenever KA630 "halts". For the KA630, "haltin means only that control is transferred to this program, not that t processor stops executing instructions.

When the console program is running, it provides services expected of VAX-11 console system. In particular, the following services a available:

- Automatic restart or bootstrap following processor halts initial power-up.
- An interactive command language allowing the user to exami and alter the state of the processor.
- Diagnostic tests executed on power up that check out the CP the memory system and the Q22-bus map.
- Support of video or hardcopy terminals as the console termin as well as support of QVSS or QDSS based bitmapped terminals.

2 ENVIRONMENT

2.1 Users

Engineering, Manufacturing, Field Service and customers will use thi program to test, configure and boot their system. This program i needed by these users both to provide an easy means to bootstrap a KA63 system and to detect and isolate system malfunctions.

Of these users, all but customers are assumed to be compute "sophisticates". While this will often be true of customers as well, n such assumption is made. Target customers include both sophisticate users who can use and understand all the features provided by th console program as well as unsophisticated users who know next t nothing about computers.

Users are not assumed to speak English. The console program may b directed to output all console messages in either English, French German, Spanish, Italian, Norwegian, Dutch, Swedish, Finish, Danish c Portuguese. If when the system powers up there is no languag specified, it prompts the user for the language to use. The use language is recorded in battery backed up RAM so the preferred languag is retained when the system is shut off.

2.2 Hardware

The console program is located in ROM on the KA630 processor board. Th ROM address range is located in the KA630's local I/O space. Th program uses the KA630 processor LEDs and console terminal output t communicate diagnostic progress and error reports to the user.

A console terminal is not required for operation but halts should not b enabled on a system without a console terminal. A QxSS (either QVSS o QDSS) based display can be substituted for the console terminal.

In order for the console program to operate, the processor must b functioning to the point of executing instructions from the consol program ROM.

The KA630 decodes the ROM addresses so that the same ROM appears mor than once in the address space. The console program is written i position independent code so that it may be executed from any addres range and the KA630 uses this feature to selectively enable and disabl the external halt circuitry. If the console program is executing in th range of 20040000 to 2004FFFF (hex), external halt conditions ar ignored. If the console program is executing in the range of 200500 to 2005FFFF (hex) external halt conditions are honored and the consol program may be "halted" (whereupon it will immediately reenter at it beginning to process the halt).

The console program normally executes from the first address range whil the diagnostics software normally executes from the second addres range.

For more information on the processor hardware, consult the KA63 Processor Specification. See also appendix B for a summary of th hardware registers used by the console program.

2.3 Software

None - the console program runs standalone, it uses no other softwar products.

2.4 Services

None - the console program requires no support services.

3 SOFTWARE CAPABILITIES

The console program is divided into six major components plus th diagnostics. These components and their responsibilities are documente

in the sections that follow. Diagnostics are documented in "KA630 RC Diagnostic Specification".

Note that throughout this document, the console program is ofte referred to simply as the "console".

The console program receives control whenever the processor halts whether becasue of an external halt signal, the execution of a hal instruction, a serious system error or because of power-up. On any hal condition, the processor switches to physical addressing, saves the PC PSL and the ISP internally, encodes and saves the cause of the halt in halt code and then branches to the start of the console program ROM Upon entry, the console program always outputs the hex value "E" to th console LED's to indicate that at least one instruction has bee executed. It then waits for power to stabilize by looping until BDR<1 is set.

When these common preliminaries are completed, the console progra checks to see if the halt is a power-up halt or a halt for some othe reason. If it is a power-up halt, it begins the power-up sequenc described in section 3.1. If it is not a power-up halt, control passe to the entry and dispatch code described in section 3.2.

3.1 Power-up

Upon power-up, the console automatically performs a variety of one tim operations. These operations initialize the processor and are neede only on power-up.

3.1.1 Power-up Mode -

On power-up, BDR<10:9> is interpreted as a power-up mode field. Th interpretation of the power-up mode field is shown in table 1. A discussed in the sections that follow, several power-up operations ar dependent on the power-up mode.

Table 1: Power-up Modes

Mode	Language Prompt	Diagnostics
0	Prompt for language only if TOY battery backup failed.	Run full diagnostics.
1	Prompt for language on every power-up.	Run full diagnostics.

Table 1 (cont.)

Mode	Language Prompt	Diagnostics
2	Set language to English.	Run console terminal loop-back tests.
3	Set language to English.	Run abbreviated diagnostics.

3.1.2 Power Stabilization And ROM Checksum Checks -

Following the application of power, the processor first perfor hardware initialization and then control is transfered to the consol program ROM code. The common console startup functions were describe previously in section 3. Following their completion, on power-up control passes to the code described here.

Beginning the power-up processing, the console code outputs the he value of "D" to the LED's indicating that the power stabilization wai is over. It then calculates a checksum of the console program ROM an checks it against the valid checksum stored in the ROM itself. If th computed checksum differs from the stored checksum, the code "hangs" i a loop. If the checksum is the same, the power-up code proceeds to th next step.

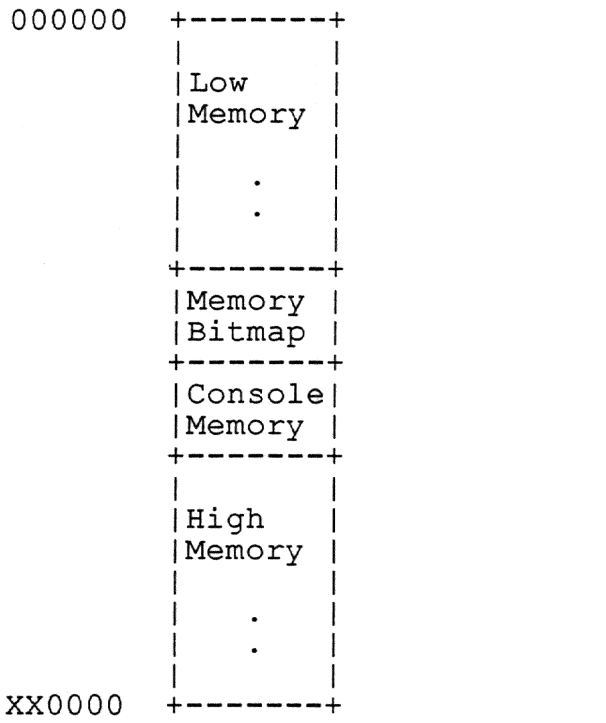
3.1.3 Console Program Initialization -

The next step of the power-up initialization is location an initialization of the memory needed for the console program itself. Th hex value of "C" is output to the LED's at the beginning of this step.

During this step, the console ROM code searches top down throug available memory for a small contiguous block that will be used by th console program for writeable storage. This block consists of two page for the console's direct use and additional pages for use to store bitmap of available memory. The amount of memory allocated to th bitmap varies according to the amount of memory available.

Following initialization, memory appears as shown in figure 1.

Figure 1: Console Memory Map



The console program memory is used by the console program for its stac and other data structures.

The bitmap is filled in later with a map of valid memory pages by th power-up memory diagnostics and this bitmap is passed to the bootstra as a map of valid memory. Beginning from the base of the bitmap, th first bit corresponds to the first page of low memory, the second bit t the second page and so on. If the bit is set, the page is good, if th bit is clear, the page failed the memory test. The bitmap does not ma itself nor any other memory that follows the bitmap. Since syste software is expected to use only pages marked as good in the bitmap, system software is not expected to modify the bitmap or the consol program memory. However, both the console program memory pages and th bitmap are checksumed by the console program to guard against accidenta modification by the system software.

If the console cannot locate enough memory for its own use and th bitmap, it "hangs".

Following initialization of its memory, the console program clears CPMBX<1:0> (halt action), CPMBX<2> (bootstrap in progress flag) an CPMBX<3> (restart in progress flag).

3.1.4 Battery Backup Check -

The next thing done by the power-up operations is to check the TOY cloc to see if the battery backup has failed. If this has happened, time c year has been lost along with the contents of all the TOY clock RAM. I the battery backup has failed, the console stops the TOY clock, zerc the time and all TOY RAM and then initializes the TOY clock control ar status registers. The operating system must check TOY clock control ar status register B and determine if the clock is stopped to know if th TOY clock contains a valid time.

No change is made to the LED's during this operation.

3.1.5 Test Interprocessor Communication Register -

Next, the interprocessor communication register is tested. The he value "B" is output to the LED's during this test.

This test is performed at this point as in addition to testing th register itself, this tests whether or not the Q22-bus is arbitratin properly. If it is not arbitrating, the processor will hang here Hanging here is preferable to hanging in the next step where we chec for QxSS video hardware on the bus. Were we to hang in the next ste without first checking the Q22-bus, the resulting LED display would b ambiguous as to which failed, the Q22-bus or the QxSS.

3.1.6 Determining The Console Terminal Type -

3.1.6.1 QxSS Determination -

Next, if the processor is an arbiter processor, the console progra checks for the presence or absence of a QxSS video display as th console device. Either a QVSS or a QDSS will be detected. If th processor is an auxiliary processor, this test is skipped. The he value "A" is output to the LED's during this test.

The QxSS is determined by testing for the presence of its CSR address a tbs. (Both QVSS and QDSS use the same CSR address.) If there is n response, no QxSS is assumed present and the console program moves t the console terminal determination code which follows. If a QxSS i detected, it is initialized and a short diagnostic is executed. If th initialization and diagnostics succeed, the console will use the QxSS a its console terminal (however, see appendix C) and skips the next ste and moves directly to the step described in section 3.1.7. If eithe the initialization or the diagnostic fails, the system hangs.

3.1.6.2 Console Terminal Determination -

If no QxSS is detected, it is assumed that normal a console terminal i connected to the console port or that NO terminal is connected. Th console program then next attempts to determined the type of termina connected. The value of "9" is output to the LED's during this test.

This determination is done by sending the terminal a device attribut request escape sequence. If the device responds with a recognizabl response, the terminal is classified as a video terminal versus a har copy terminal based on that response. This information is used when i console I/O mode to govern how command line editing is performed. Th terminal must respond in 1 second to the device query. If no respons or the response is not recognized, the test is repeated two more time or until the response is understood. If the device never responds c the response isn't recognized, the terminal is classified as a hard cop terminal.

Terminal response will be recognized in either eight bit or seven bi mode.

In addition to determining the presence or absence of a recognizable CR console, this information is also used to determine if the termina supports the DEC multinational character set (MCS). The console progra assumes that all new terminals, VT2xx and beyond support MCS. If th terminal does not support MCS, CPMBX<7:4> is set to 2, forcing Englis as the console display language.

3.1.7 Console Message Language Check -

The console next outputs a hex value of "8" to the LED's and the determines the appropriate language to use for all console messages The algorithim used to determine the language appears below. Th console language is stored in CPMBX<7:4> (see appendix B).

1. If power-up mode (see table 1) is 2 or 3, set the consol language to English and exit.

2. If power-up mode is 0 or if the value of CPMBX<7:4> is zero solicit the language from the user. If the user doesn' respond within 30 seconds, set the language to English (3) an exit.

Remember that when the terminal is queried, if it isn't recognized a one that supports MCS, CPMBX<7:4> is set to 2, forcing English as th console language. English messages use the 7 bit subset of MCS. Also, if a loss of power for the TOY clock is detected, the contents of th TOY RAM is zeroed meaning that step 2 of the algorithm above will caus the user to be prompted.

The format of the user query issued in step 2 is shown in Figure 2.

Figure 2: User Language Prompt

```
    1) Danish              7) Dutch
    2) German              8) Norwegian
    3) English             9) Portuguese
    4) Spanish            10) Finish
    5) French             11) Swedish
    6) Italian

  (1-11):
```

The text will be displayed using the DEC Multinational Character se extension.

The user must respond by entering a value of 1 through 11 following th question mark prompt, any other input will be rejected and the promp line will be issued again. Normal console input line editing feature (see section 3.6.1) will be available during input.

If the console program has determined that a QxSS video display syste is being used as the console, an additional step is required. The QxS display system uses the DEC LK201 keyboard which comes in 16 nationa variants. The keyboard variant cannot be determined by querying th keyboard itself, it must be determined either from the language selecte or by means of an additiona menu selection. If French, German o English is specified as the language, the keyboard variant is ambiguou and the appropriate menu from figure 3 is displayed and the user i prompted to specify the which national keyboard variant is in use. I the user does not respond in 30 seconds, the last selection is assumed.

Figure 3: QxSS Keyboard prompts

```
----------French----------

      1) Canada
      2) France/Belgium
      3) Switzerland

      (1..3):

----------German----------

      1) Germany/Austria
      2) Switzerland

      (1..2):

----------English----------

      1) United Kingdom
      2) United States/Canada

      (1..2):
```

3.2 Entry/Dispatch

Following the determination of the console language on power-up, c directly on entry from any other halt condition, the console dispatche to the appropriate code to service the halt.

To determine what actions to take, the console program examines the ha code, the halt enable bit (BDR<14>), the console program mailbo register (CPMBX) and then acts in accordance with decision table show in table 2.

Table 2: Console Entry Decision Table

BDR(14)	Power-up	CPMBX(1:0)	Actions
F	T	X	Diagnostics, bootstrap, halt.
F	F	0	Restart, bootstrap, halt
F	F	1	Restart, halt.
F	F	2	Bootstrap, halt.
F	F	3	Halt
T	T	X	Diagnostics, halt.
T	F	0	Halt.
T	F	1	Restart, halt.
T	F	2	Bootstrap, halt.
T	F	3	Halt.

A power-up entry is one where the processor halt code is 3. "X" mean "don't care" and the meanings of the other CPMBX codes are defined i appendix B, section B.2.1.

Multiple actions in table 1 mean that the first action is taken and i and only if it fails, the next action is taken. Diagnostics are a exception. If diagnostics fail the console will "hang" witho attempting to bootstrap the processor. If they succeed, then the nex action is taken.

Note that because the KA630 does not support battery backup, it examine the halt code and does not attempt to perform restart operation following power-up.

3.3 Diagnostics

On power-up, the Entry/Dispatch code dispatches first to the diagnostic to check out the processor and memory before proceeding. Before doin this, the console outputs the message "Performing normal system tests. to the console terminal. As each test is run, a code is displayed o the processor LED's and output to the console terminal as well, causin a "countdown" to be displayed on both.

The first diagnostic LED code is 8 so executing the diagnostic continues the LED countdown. The diagnostic codes are documented i table A-1 along with all other codes. See figures 9 and 10 in section for console display formats.

At the conclusion of all tests, the message "Tests completed." is outpu to the console terminal.

If a diagnostic test detects a fatal error, an error message i displayed on the console, a summary message is displayed indicating tha the continued operation is not possible and then the system "hangs" leaving the test code in the LED's. If halts are disabled, the only wa